Cellular Network Infrastructure - Support #1965

use sysmoUSIM-SJS1 with 3G OsmoMSC

03/04/2017 03:18 AM - neels

Status:	Closed	**Start date:**	03/04/2017
Priority:	High	**Due date:**	
Assignee:	neels	**% Done:**	100%
Category:			
Target version:			
Spec Reference:			

Description

A stock sysmoUSIM-SJS1 seems to refuse to authenticate with my 3G OsmoMSC.
I have a second sysmoUSIM that dexter reprogrammed to EF.AUTH = 0101, i.e. Milenage for both GSM and UMTS, which works well.
Clarify whether the stock sysmoUSIM can be made to work with 3G without reprogramming it,
i.e. by fixing the OsmoMSC / OsmoHLR code instead of adjusting the sysmoUSIM.
(Interesting for the accelerate3g5 folks)

Related issues:

Related to OsmoHLR - Feature #1969: add IND for proper UMTS auth resync	**Closed**	**03/07/2017**
Related to libosmocore - Bug #1968: upon auth resync with osmo_auth_gen_vec_a...	**Closed**	**03/07/2017**
Related to OsmoNITB - Feature #1711: 3G Auth	**Closed**	**05/14/2016**

History

#1 - 03/04/2017 10:15 AM - laforge

On Sat, Mar 04, 2017 at 03:18:29AM +0000, neels [REDMINE] wrote:

> A stock sysmoUSIM-SJS1 seems to refuse to authenticate with my 3G OsmoMSC.

the big question is why.

> Clarify whether the stock sysmoUSIM can be made to work with 3G without reprogramming it. (Interesting for the accelerate3g5 folks)

there are plenty of customers that use the sysmoUSIM-SJS1 with 3G and LTE networks of various vendors, so I don't think there's anything wrong in general.

Also, we know from manual verificationw with osmo-sim-auth that the procedures on the SIM cards work. There must be something in the Osmocom 3G coee that breaks it, or something is wrong/odd about the specific card you use.

You can always use simtrace to check what's happening between card and pone. Also, you can use osmo-sim-auth to dry-test the card.

Slightly unrelated In fact, it might even be fun to have something lik eosmo-sim-auth that talks GSUP to the HLR to obtain authentication vectors, bypassing MS, UTRAN and MSC.

#2 - 03/04/2017 03:32 PM - neels

laforge wrote:

> the big question is why.

Exactly. Didn't come around to that yet, next on my agenda.

> Slightly unrelated In fact, it might even be fun to have something lik eosmo-sim-auth that talks GSUP to the HLR to obtain authentication

vectors, bypassing MS, UTRAN and MSC.

hehe nice, and maybe osmo-auc-gen could access the HLR as well (though that would remain mostly manual anyway and adding the dep may not really be worth it)

#3 - 03/04/2017 03:33 PM - neels

- Description updated

#4 - 03/04/2017 07:54 PM - neels

- % Done changed from 0 to 30

Reprogrammed the not-working USIM to EF.AUTH = 0101, and the same failure occurs.
I see that both SGSN and MSC attempt a resync at the same time, first suspected that this causes problems.

Switching off HLR in SGSN, setting to auth-policy accept-all, and testing again...

I notice that the HLR generates 5 keys after AUTS and the MSC sends out the **second** key instead of the first newly generated one, which is of course a VLR bug.
I assumed that this off-by-one would be in the range of acceptable SQN mismatch (which allows MSC and SGSN to use keys with different SQN at the same time), but apparently this is stricter during resync?

Remaining to test is a simultaneous resync of MSC and SGSN -- if the SQN is indeed this sensitive, that could be a problem.
I thought I had already tested this (by altering the SQN in the DB) but apparently not well enough.

#5 - 03/04/2017 08:01 PM - neels

Indeed, when I change the unit test for AUTS to return two instead of just one auth vector, the test fails because the second vector is sent for auth instead of the first. Fixing...

#6 - 03/04/2017 08:27 PM - neels

- Status changed from New to In Progress

With that fixed, things are still not working out.
Also adjusted the in-db SQN to match the SQN the USIM indicates, which should avoid AUTS resync altogether, but still doing AUTS and failing.

I'll check every last bit everywhere now -- the keys stored on the USIM and HLR.
Though, the fact that the AUTS received from the USIM produces a sane SQN number (i.e. 0) should mean that keys are correct.
Maybe some obscure detail is causing AUTS to fail over 3G? Will test over 2G with this USIM as well.

#7 - 03/04/2017 11:35 PM - neels

It turns out, from using the sysmo-usim-tool, that the OP/c written on the USIM doesn't match what is listed as OPC in our USIM database.

There are social questions mixing with my lack of knowledge: has dexter fixed the OPC on the card he gave me (meaning that other sysmoUSIMs would also need fixing)? Has the OPC been overwritten on the cards I got from Harald? Does the one from dexter store OPC while the other two store OP and should actually be correct?

To summarize, I have 3 sysmoUSIMs on my desk.
To avoid confusion with 3G issues, testing on an R99 GSM network (sysmoBTS):

IMSI	full UMTS auth works	tried with EF.AUTH value	"OP" value shown	"OP/OPc" value shown	got it from
..10650	no	0x0101	0x0	mismatches "OPC" in sysmocom's SIM card db, but same value as 10651	Harald
..10651	no	0x0101	0x0	mismatches, but same value as 10650	Harald
..10652	yes	0x0101	0x1	matches "OPC" in sysmocom's SIM card db	dexter

reading OP and OPC with:

```
./sysmo-usim-tool.sjs1.py -a <adm1> -o
```

So I assume 0x1 for "OP" value means an OPC is stored on the SIM, so the not working ones store an OP which makes sense since both are identical.

When I store the OPC from our sim card database on the 10651, it reads back as 0x1 with the OPC I wrote, but authentication *still* doesn't work.

Browsing the other sysmo-usim-tool parameters to find mismatches between the cards...

damn, now I used the wrong adm key on the working 10652 USIM a number of times and it refuses to show me any more details. I guess now
"The number of attempts to enter the ADM1 key was exceeded. At this point, the card cannot be recovered and you will never be able to authenticate using ADM1 key again."
Gah, I could kick myself! Doesn't it have some sort of PUK? how can this be so fragile!

The other USIMs show the default milenage params as seen in the sysmo-usim-manual.pdf,
but since I locked myself out of the working one I won't be able to look what *its* parameters are.

So far I can't verify the KI because pySim-read.py can't read from the sysmoUSIM-SJS1.
I kind of assume KI and OP were actually correct because the AUTS calculations yielded a sensible sequence nr...

So unfortunately I still don't know why one USIM works and the other two don't.
At least now I know that it also happens identically on an R99 GSM, so the failure is not related to the 3G code.

#8 - 03/05/2017 12:03 AM - neels

neels wrote:

> So I assume 0x1 for "OP" value means an OPC is stored on the SIM

No need to assume, the sysmo-usim-manual clearly states so.

I extended the sysmo-usim-tool to output the KI value here , and indeed can confirm the correct KIs as found in sysmocom's card database on both cards that aren't authenticating.

#9 - 03/05/2017 12:36 AM - neels

Verified with card reader and osmo-sim-auth.py that the 10651 keeps sending the same AUTS to sync to SQN 1 over and over:

```
▶ osmo-auc-gen -3 -a milenage -k C9BEB329BA89F5F106911BA5D614E401 -o 6DC412B006B67D49CA696A064E618EB0
osmo-auc-gen (C) 2011-2012 by Harald Welte
This is FREE SOFTWARE with ABSOLUTELY NO WARRANTY

WARNING: We're using really weak random numbers!

RAND:    a598420d89b6765583d6885bc37aaf11
AUTN:    524e49357561000039c37e8bfa9425c9
IK:    5eb7f484ebc34eb08a371d3cf882943e
CK:    b9c1fff643c5f13e31e375302ea76b52
RES:    e4d6aba046ff40d6
SRES:    a229eb76
Kc:    5ca2637e7e2340e2

▶ ./osmo-sim-auth.py -r a598420d89b6765583d6885bc37aaf11 -a 524e49357561000039c37e8bfa9425c9
[+] UICC AID found:
found [AID 1] 3GPP || USIM || (255, 255) || (255, 255) || (137, 7, 9, 0, 0)
[+] USIM AID selection succeeded

Testing USIM card with IMSI 901700000010651

UMTS Authentication
AUTS:    01416a43f517dc078a73aa9f0593

GSM Authentication
SRES:    a229eb76
Kc:    5ca2637e7e2340e2

▶ osmo-auc-gen -3 -a milenage -k C9BEB329BA89F5F106911BA5D614E401 -o 6DC412B006B67D49CA696A064E618EB0 -r a5984
20d89b6765583d6885bc37aaf11 -A 01416a43f517dc078a73aa9f0593
osmo-auc-gen (C) 2011-2012 by Harald Welte
This is FREE SOFTWARE with ABSOLUTELY NO WARRANTY
```

```
RAND:    a598420d89b6765583d6885bc37aaf11
AUTN:    524e49357560000007ba593f416804ef
IK:    5eb7f484ebc34eb08a371d3cf882943e
CK:    b9c1fff643c5f13e31e375302ea76b52
RES:    e4d6aba046ff40d6
SRES:    a229eb76
Kc:    5ca2637e7e2340e2
AUTS success: SQN.MS = 0, generated vector with SQN = 1, next SQN = 2

▶ ./osmo-sim-auth.py -r a598420d89b6765583d6885bc37aaf11 -a 524e49357560000007ba593f416804ef
[+] UICC AID found:
found [AID 1] 3GPP || USIM || (255, 255) || (255, 255) || (137, 7, 9, 0, 0)
[+] USIM AID selection succeeded

Testing USIM card with IMSI 901700000010651

UMTS Authentication
AUTS:    01416a43f517dc078a73aa9f0593

GSM Authentication
SRES:    a229eb76
Kc:    5ca2637e7e2340e2

▶ osmo-auc-gen -3 -a milenage -k C9BEB329BA89F5F106911BA5D614E401 -o 6DC412B006B67D49CA696A064E618EB0 -r a5984
20d89b6765583d6885bc37aaf11 -A 01416a43f517dc078a73aa9f0593
osmo-auc-gen (C) 2011-2012 by Harald Welte
This is FREE SOFTWARE with ABSOLUTELY NO WARRANTY

RAND:    a598420d89b6765583d6885bc37aaf11
AUTN:    524e49357560000007ba593f416804ef
IK:    5eb7f484ebc34eb08a371d3cf882943e
CK:    b9c1fff643c5f13e31e375302ea76b52
RES:    e4d6aba046ff40d6
SRES:    a229eb76
Kc:    5ca2637e7e2340e2
AUTS success: SQN.MS = 0, generated vector with SQN = 1, next SQN = 2

▶ ./osmo-sim-auth.py -r a598420d89b6765583d6885bc37aaf11 -a 524e49357560000007ba593f416804ef
[+] UICC AID found:
found [AID 1] 3GPP || USIM || (255, 255) || (255, 255) || (137, 7, 9, 0, 0)
[+] USIM AID selection succeeded

Testing USIM card with IMSI 901700000010651

UMTS Authentication
AUTS:    01416a43f517dc078a73aa9f0593

GSM Authentication
SRES:    a229eb76
Kc:    5ca2637e7e2340e2
```

while the same dance works with the 10652 card:

```
▶ osmo-auc-gen -3 -a milenage -k 92D6A839C5BA10BE6EBA85B0C7265783 -o 9B994AFC213377798E06A4D86C2E27D4
osmo-auc-gen (C) 2011-2012 by Harald Welte
This is FREE SOFTWARE with ABSOLUTELY NO WARRANTY

WARNING: We're using really weak random numbers!

RAND:    a319a864af9c8028d2b12036e5f3077a
AUTN:    ae0dc8675f28000087f19005eaed5492
IK:    eb6443096347999693dfc914fdd1f711
CK:    ada8a9d28e467a38c58047409ee16b56
RES:    81bdb44f55e29665
SRES:    d45f222a
Kc:    1093648f8e317fe9

▶ ./osmo-sim-auth.py -r a319a864af9c8028d2b12036e5f3077a -a ae0dc8675f28000087f19005eaed5492
[+] UICC AID found:
found [AID 1] 3GPP || USIM || (255, 255) || (255, 255) || (137, 7, 9, 0, 0)
[+] USIM AID selection succeeded

Testing USIM card with IMSI 901700000010652

UMTS Authentication
```

```
AUTS:    a1f41984cb6bd13400b87a533c5c

GSM Authentication
SRES:    d45f222a
Kc:    1093648f8e317fe9

▶ osmo-auc-gen -3 -a milenage -k 92D6A839C5BA10BE6EBA85B0C7265783 -o 9B994AFC213377798E06A4D86C2E27D4 -r a319a
864af9c8028d2b12036e5f3077a -A a1f41984cb6bd13400b87a533c5c
osmo-auc-gen (C) 2011-2012 by Harald Welte
This is FREE SOFTWARE with ABSOLUTELY NO WARRANTY

RAND:    a319a864af9c8028d2b12036e5f3077a
AUTN:    ae0dc8675e4b0000c863e82c5d44078a
IK:    eb6443096347999693dfc914fdd1f711
CK:    ada8a9d28e467a38c58047409ee16b56
RES:    81bdb44f55e29665
SRES:    d45f222a
Kc:    1093648f8e317fe9
AUTS success: SQN.MS = 354, generated vector with SQN = 355, next SQN = 356

▶ ./osmo-sim-auth.py -r a319a864af9c8028d2b12036e5f3077a -a ae0dc8675e4b0000c863e82c5d44078a
[+] UICC AID found:
found [AID 1] 3GPP || USIM || (255, 255) || (255, 255) || (137, 7, 9, 0, 0)
[+] USIM AID selection succeeded

Testing USIM card with IMSI 901700000010652

UMTS Authentication
RES:    81bdb44f55e29665
CK:    ada8a9d28e467a38c58047409ee16b56
IK:    eb6443096347999693dfc914fdd1f711
Kc:    1093648f8e317fe9

GSM Authentication
SRES:    d45f222a
Kc:    1093648f8e317fe9

▶
```

So, KI and OPC are correct on the cards (specifically wrote 0x1 plus OPC and read the KI), the algos for 2G and 3G are both set to milenage, and I'm doing everything right as far as one of the USIMs is concerned. So there must be something I don't know that causes the other USIMs to refuse.

#10 - 03/05/2017 09:36 AM - laforge

On Sat, Mar 04, 2017 at 11:35:35PM +0000, neels [REDMINE] wrote:

> It turns out, from using the
> sysmo-usim-tool, that the
> OP/c written on the USIM doesn't match what is listed as OPC in our
> USIM database.

Well, the first question is whether or not a given card is using OP or OPC, and whether the configuration in the HLR matches this for the given card ;)

> damn, now I used the wrong adm key on the working 10652 USIM a number of times and it refuses to show me any more details. I guess now
> "The number of attempts to enter the ADM1 key was exceeded. At this point, the card cannot be recovered and you will never be able to authenticate using ADM1 key again."
> Gah, I could kick myself! Doesn't it have some sort of PUK? how can this be so fragile!

It is not fragile at all, it is the most expected behavior: If you are tinkering with the operator key to re-program a card, then you should know what you are doing. If the key is entered wrong the specified number of times, the card is permanently unrecoverable. The ADM is the **ultimate** key, there is no higher key. It is a very good idea tha there's a very limited amount of attempts. As an operator, you don't want any random unauthorized person to have an unlimited number of attempts to try to break into your card.

Any software should immediately tell you on a failed attempt

- that the reason for the failure was caused by wrong ADM key
- how many tries you have left.

If either pySim-prog or osmo-usim-tool don't do that, they should be modified to clearly inform the user about this.

> I kind of assume KI and OP were actually correct because the AUTS calculations yielded a sensible sequence nr...

As indicated, the fundamental question to me is whether you intend to be working with OP or OPc, and whether the configuration is identical on both the HLR and the USIM side.

#11 - 03/06/2017 01:01 AM - neels

laforge wrote:

> Well, the first question is whether or not a given card is using OP or OPC, and whether the configuration in the HLR matches this for the given card ;)

When the AUTS calculations make sense, it's a pretty sure sign that things match.
Later on they definitely matched and it was still not working.

> > Gah, I could kick myself! Doesn't it have some sort of PUK? how can this be so fragile!
>
> It is not fragile at all, it is the most expected behavior

Yes, I was just angry at myself to call an erratic script that does 5 things in a row with the wrong adm1 key, so that I forfeited it with a single keypress without being able to back out in time.

So I wished there was a mechanism to e.g. wait 5 minutes and have another try.
This way I'm locked out just because I hit enter once...
My fault but would be nice to not be locked out permanently.

> Any software should immediately tell you on a failed attempt
>
> - that the reason for the failure was caused by wrong ADM key
> - how many tries you have left.

sysmo-usim-tool basically complains about a failure.
IIRC it's from the UI not distinguishable whether there are N tries left or the adm1 is already locked down.

> As indicated, the fundamental question to me is whether you intend to be working with OP or OPc, and whether the configuration is identical on both the HLR and the USIM side.

I assumed since OPC is derived from OP and KI that it doesn't matter much which is stored as long as the USIM knows which one it has. But anyway:

Yes, I programmed the 10650 and 10651 cards to a) use OPC and b) programmed exactly the OPC from the HLR db,
and c) confirmed that the KI is exactly identical. Still got the AUTS reiterating to sync back to SQN.MS=0 every time over.
Also the cmdline transcript seen above uses exactly the OPC (not OP) and KI confirmed to be on the card.

Looking forward to find out the detail that causes the failure, I hope it's not my stupidity but at this point I'm fairly sure it's just ignorance of some detail.
Maybe dexter set some parameter on the 10652, looking forward to ask him tomorrow.
I'm really glad the 10652 works and worked, I would still be looking for bugs in my code had I tested with the other two USIMs from the start...

#12 - 03/07/2017 02:10 PM - neels

- % Done changed from 30 to 100

took a brand new USIM and tried using osmo-sim-auth cmdline tool and a card reader.
It behaves the same way as the two non-working USIMs: keeps sending an AUTS to resync to SQN.MS = 0.
Trying to use OP lead me to #1967.

But then Harald reminded me of the fact that the lower $nr of bits of the SQN are irrelevant for resync.
A USIM has an IND setting indicating the number of bits that are irrelevant, the usual value seems to be 5.
That would mean that when syncing to SQN.MS = 0, we need to next use at least SQN = 2^5 = 32.

Indeed using osmo-auc-gen to produce AUTN with SQN = 32 leads to successful authentication!

The result is that our osmo_auth code should, upon resync, add 2^IND to SQN.MS --> #1968#1969

The SIM card that is working despite SQN+1 increments possibly is configured with a different IND,
or maybe during all resync tests a bit above the 2^IND dead area was flipped by pure coincidence.

#13 - 03/07/2017 03:13 PM - neels

Checked again, and the working USIM does often accept SQN+1 even if it does not flip a 1<<IND or above bit.

Also the USIM that was initially not working is now accepting AUTS resync with SQN increments below the 1<<IND range.

Possibly the USIM needs to generate new accepted SQNs at first and once SQNs are generated in the pool, using these will work even if a resync happened.
i.e. if I provoke a resync by sending an AUTN from SQN = 33 a second time, then using SQN = 34 for the next attempt will work.

It seems that a larger SQN jump is only needed at the very first attempt, when SQN.MS is still zero.

I also tried to provoke an error by "jumping" to SQN=93, just below the 2^5 threshold of 96, and then causing an AUTS resync.
After that I would expect 94 and 95 to not be valid, but 96 to work again since it flips the 1<<5 bit.
However, all of those SQNs are accepted.

The restriction doesn't really seem to have an effect beyond the very first "bootstrapping" of UMTS auth on the sysmoUSIM-SJS1.
I was probably just lucky when trying with the 10652 USIM, in that the HLR DB's SQN had advanced far enough because of testing.

The restriction is very hard to hit once I did the very first jump from 0 to 32.
I can even authenticate with an SQN 129, then jump back to SQN 51:

```
0.............32.............64.............96.............128.........
 [never works]
               ^1.works
                                                                ^2.jump to 129 works
                        ^3.then back to 51 works
```

An AUTS resync will always jump to the last used one, e.g. if I jumped to a high number like 129, AUTS will reflect SQN.MS=129.
But lower SQN will still work as long as they have not been used before.
At some point the lower SQN will cause an AUTS, but the sysmoUSIM-SJS1 seems to have way more than 32 slots for valid SQN.

Basically, "everything works all the time", except 0..31 never work.
I can also take a fresh USIM, and directly generate a key with SQN=32, which will work right away.
No AUTS resync process is necessary, all will work out as long as the first SQN used with a sysmoUSIM-SJS1 is >= 32.

I can **even** take a fresh sysmoUSIM-SJS1 and start off with SQN = 4096 right away.
After that SQN 3073 does also work, but 3072 causes an AUTS resync.

So the sysmoUSIM-SJS1 seems to behave as follows:

- SQN 0-31 never work.
- At any time, any larger SQN with as high a number as desired works right away (at least I know that 4096 higher than last time works).
- Once such a high SQN X has been used, SQNs down to and including X - 1023 also still work.
- SQN <= X - 1024 cause an AUTS resync.
- No SQN can be re-used when it has been used once, its unused neighbors still work.

So there seems to be a buffer for 1024 SQN backwards from whichever SQN was last used.
Makes sense if one imagines that IND could be chosen as 10, in which case the USIM would need to store 1024 SQNs.
It seems to do so even though IND is chosen smaller.

Basically all we need to do to get sysmoUSIM-SJS1 working with OsmoHLR is to have a default SQN of 32 in the database.

#14 - 03/07/2017 03:17 PM - neels

- Status changed from In Progress to Resolved

#15 - 03/07/2017 03:51 PM - neels

neels wrote:

> So there seems to be a buffer for 1024 SQN backwards from whichever SQN was last used.

According to tests, this buffer also persists across sysmoUSIM-SJS1 power-cycles, making it really hard to hit the IND limitation, *except* when first using the sysmoUSIM-SJS1 (with an SQN < 32).

#16 - 03/07/2017 06:26 PM - laforge

Hi Neels,

did you study Annex C.2 of 3TPP TS 33.102? This documents the intended behavior of a USIM. As far as I can tell, in chronological order, the following checks are made:

C.1:

let SQN = SEQ | IND

- if SEQ (received) - SEQ > Delta: generate AUTS
  - delta value in sysmoUSIM is TBD. Common values (see Profiles of C.3) are 2^28 or 2^24

SQN is the value used in (encrypted) transmission
SEQ and IND are values only used internally inside USIM and AUC

C.2:

Step a (optional)

- if the SIM has a Limit L configured, and
- if SEQ_MS(highest ever accepted) - SEQ < L: proceed
- if SEQ_MS(highest ever accepted) - SEQ >= L: AUTS

Step b (mandatory)

- the USIM keeps an array of size IND
- the USIM stores the last **accepted** SEQ for each IND in this array and calls it SEQ_MS(i)
- if SEQ (received) > SEQ_MS(i): Success (proceed) and update SEQ_MS(i)
- if SEQ (received) > SEQ_MS(i): generate AUTS

Finally, in C.3.4 it is specified how a AUC shall behave.

- in general, IND shall be incremented modulo the ind-size
- but, there are "useful exceptions" where basically all vectors within a given batch (i.e. a single SendAuthInfo) shall have identical IND (and thus increasing SEQ), and where CS/PS domains shall receive different IND values.

On Tue, Mar 07, 2017 at 03:13:47PM +0000, neels [REDMINE] wrote:

> Checked again, and the working USIM does often accept SQN+1 even if it does not flip a 1<<IND or above bit.

yes, that's how it should be. Only in "rare" cases, re-sync shall be performed.

> Also the USIM that was initially not working is now accepting AUTS resync with SQN increments below the 1<<IND range.

ok.

> Possibly the USIM needs to generate new accepted SQNs at first and once SQNs are generated in the pool, using these will work even if a resync happened. i.e. if I provoke a resync by sending an AUTN from SQN = 33 a second time, then using SQN = 34 for the next attempt will work.

possibly.

> I also tried to provoke an error by "jumping" to SQN=93, just below the 2^5 threshold of 96, and then causing an AUTS resync. After that I would expect 94 and 95 to not be valid, but 96 to work again since it flips the 1<<5 bit. However, all of those SQNs are accepted.

I think all those jumps are way to small to trigger any of the protection mechanisms of Annex C.2

> An AUTS resync will always jump to the last used one, e.g. if I jumped
> to a high number like 129, AUTS will reflect SQN.MS=129.

yes, that's in-line with the spec

> But lower SQN will still work as long as they have not been used before.

yes, as using them will increment the SEQ_MS(i) for the given IND, i.e.
the "array bucket for this IND has been used.

> At some point the lower SQN will cause an AUTS, but the sysmoUSIM-SJS1
> seems to have way more than 32 slots for valid SQN.

It depends on what was the last SEQ_MS(i) for that given bucket

> Basically, "everything works all the time", except 0..31 never work.
> I can also take a fresh USIM, and directly generate a key with SQN=32, which will work right away.
> No AUTS resync process is necessary, all will work out as long as the
> first SQN used with a sysmoUSIM-SJS1 is >= 32.

I think it is a logical consequence of C.2.2 (b):

"If SEQ ≤ SEQ MS (i) the USIM shall generate a synchronisation failure
message using the highest previously accepted sequence number anywhere
in the array, i.e. SQN MS ."

SEQ_MS(i) must be zero when the card is new. So if the USIM receives an
auth request with SEQ=0, then "0 <= 0" and hence an AUTS is generated.

However, if the initial auth request has SEQ=1 or higher, then "1 > 0"
and the authentication proceeds.

> I can **even** take a fresh sysmoUSIM-SJS1 and start off with SQN = 4096 right away.

yes, as 4096 is much smaller than the L which is presumably 2^24 or
2^28.

> After that SQN 3073 does also work, but 3072 causes an AUTS resync.

yes, because 4096%32 is 0 and 3072%32 is also 0, so it maps into the
same "array bucket", and you cannot have SEQ going backwards inside one
bucket.

#17 - 03/08/2017 02:40 AM - neels

laforge wrote:

> did you study Annex C.2 of 3TPP TS 33.102?

One of those tremendously useful Annexes =)

You're right of course. Instead of remembering 1024 SQNs, the sysmoUSIM does as described in the annex.
The lower significant part (i.e. SQN % (1<<IND)) gives an array index, and the SQNs used for that array index must not go backwards;
verified to be the case with a sysmoUSIM just now. I was just "lucky" before.

So we should probably hand out SQNs to MSC and SGSN by incrementing each SQN with 1<<IND?
So that e.g. the MSC gets SQNs 32, 64, 96, 128, 160, and the SGSN gets SQNs 33, 65, 97, 129, 161?
Then each is enforced to stay in sequence in the SQN "bucket"/IND-slot it received. Does that make sense?

Such IND-slot actually spans the entire SQN value range.

Then with an IND of 5 we can have 32 HLR clients with each getting their individual IND-slot,
which means choosing IND is an intrinsic network distribution design choice!
If we encounter a USIM having less IND slots than we have HLR clients, stuff breaks down.

We should deal out SQN IND-slots by HLR client index.

Maybe to get the same buckets after reconnection, sort the clients by id...
or hash the id to a number 0..31, with a special check against collision... nah whatever...

It's not very harmful for clients to switch/swap over to another slot,
it will maybe cost one AUTS procedure and will then carry on as usual.
HLR clients wouldn't be expected to crash and re-establish that often ... right?

When an AUTS is received, we want to stay within that client's IND-slot and hence increment by 1<<IND.

So ... do we also store each IND-slot's highest SQN individually in the HLR DB?
So far we store only one last SQN, causing the "slower" clients to skip forward with the "fastest" client.
Technically the MSC could still be at 160 while the SGSN is already at 8193 (also verified with sysmoUSIM).

...

loosely related detail:
"Using the above array mechanism, it is not required that a previously visited VLR/SGSN deletes the unused authentication vectors when a user de-registers from the serving network (super-charger concept). Retaining the authentication vectors for use when the user returns later may be more efficient as regards signalling when a user abroad switches a lot between two serving networks."

Which basically means we could/should decide to keep a vlr_subscr in RAM if it still has valid auth tuples.
Which means we would practically keep all VLR subscribers -- which we want to *avoid*.
But the switching-back-and-forth seems like a sensible thing to care about, so we might want to add a timeout before deallocating.

#18 - 03/08/2017 02:45 AM - neels

neels wrote:

> Technically the MSC could still be at 160 while the SGSN is already at 8193 (also verified with sysmoUSIM).

...until the next AUTS occurs, in which case the AUTSing client's SQN catches up with whichever client was the fastest.
And this doesn't happen very often.

#19 - 03/08/2017 09:15 AM - laforge

Hi Neels,

On Wed, Mar 08, 2017 at 02:40:29AM +0000, neels [REDMINE] wrote:

> So we should probably hand out SQNs to MSC and SGSN by incrementing each SQN with 1<<IND?

In other words: Incrementing SEQ, not SQN.

> So that e.g. the MSC gets SQNs 32, 64, 96, 128, 160, and the SGSN gets
> SQNs 33, 65, 97, 129, 161? Then each is enforced to stay in sequence
> in the SQN "bucket"/IND-slot it received. Does that make sense?

yes. That's one of the methods/exceptions specified in said annexes.

> Then with an IND of 5 we can have 32 HLR clients with each getting their individual IND-slot,
> which means choosing IND is an intrinsic network distribution design choice!
> If we encounter a USIM having less IND slots than we have HLR clients, stuff breaks down.

well "breaks down" means we see more AUTS, so it's just an optimization
to make AUTS less likely that "breaks down".

> Maybe to get the same buckets after reconnection, sort the clients by
> id... or hash the id to a number 0..31, with a special check against
> collision... nah whatever...

I think we shouldn't spend too much time to be tricky here for now.
Hashing the ID (which might be a stringified GT later on) should be
sufficient. When there are hash collissions our optimiziation is
slightly less optimal, but we don't care about this until it becomes a
problem, if ever.

> HLR clients wouldn't be expected to crash and re-establish that often ... right?

no, not really - but even if they do, they should normally re-connect
using the same GT and hence end up in the same bucket.

> When an AUTS is received, we want to stay within that client's
> IND-slot and hence increment by 1<<IND.

Yes, SEQ should be incremented, not SQN. But I think the specs (33.102
Sect. 6.3.5) state it quite clearly what needs to be dne, i.e., we have
to use the SQN provided in the AUTS.

"If the verification is successful the HE/AuC resets the value of the
counter SQN HE to SQN MS."

Since the MS has a duty to store the highest ever received SQN in any
authentication procedure (SQN_MS), it is sending that SQN as part of the
AUTS request, and the HLR simply "trusts" this authenticated number.

It doesn't matter if we previously might have produced vectors with
higher SQN and they are stored somewhere in some VLR or SGSN (or already
deleted). As long as they never made to the USIM, the SQN_MS is not
incremented to their value, and hence it only matters which SQN were
ever received/seen by the USIM, not which were the largest ever
generated by the HLR/AUC.

> So ... do we also store each IND-slot's highest SQN individually in the HLR DB?

I don't see why, and the specs don't seem to suggest that.

> So far we store only one last SQN, causing the "slower" clients to
> skip forward with the "fastest" client. Technically the MSC could
> still be at 160 while the SGSN is already at 8193 (also verified with
> sysmoUSIM).

And where is the problem with that? As long as the values for the MSC
IND bucket have not been used, they are still accepted, even if they're
numerically much lower.

> "Using the above array mechanism, it is not required that a previously
> visited VLR/SGSN deletes the unused authentication vectors when a user
> de-registers from the serving network (super-charger concept).
> Retaining the authentication vectors for use when the user returns
> later may be more efficient as regards signalling when a user abroad
> switches a lot between two serving networks."
>
> Which basically means we could/should decide to keep a vlr_subscr in
> RAM if it still has valid auth tuples.

yes.

> Which means we would practically keep all VLR subscribers -- which we
> want to *avoid*.

well, it could be a later decision to simply introduce a memory limit
(or quantity limit) up to which to keep detached subscribers in the VLR
as an optimization. A fixed-max-size VLR table seems to be what
large-scale VLRs use in production out there.

All it would take in our implementation is a move away from the linear
list to a hash table or a tree, together with a user-configurable size
for the table.

But once again, this is not our immediate concern now.

> But the switching-back-and-forth seems like a sensible thing to care
> about, so we might want to add a timeout before deallocating.

Sure, if you are moving between cells on a VLR boundary, it is likely
you switch back and forth, just like you would on aboundary between two

LACs within a NITB right now. You don't even need to move. Even for a stationary MS/UE, just the radio propagation changes have to exceed the cell reselection hysteresis and then you have your switch.

Once again, not our primary concern now, as we first have to validate a setup with multiple VLRs and one OsmoHLR in the first place, and our networks have to become large enough to care about this. But definitely worth adding some feature / wishlist tickets at low priority.

#20 - 03/08/2017 03:14 PM - neels

laforge wrote:

> Then with an IND of 5 we can have 32 HLR clients with each getting their individual IND-slot,
> which means choosing IND is an intrinsic network distribution design choice!
> If we encounter a USIM having less IND slots than we have HLR clients, stuff breaks down.

well "breaks down" means we see more AUTS, so it's just an optimization to make AUTS less likely that "breaks down".

Yes, it will still work, but the scheme of dealing out of tuples will become very inefficient, placing the calculation burden of lots of unused auth tuples on the HLR.

It plays out like this; for the sake of the example, let me write SQN as SEQ_IND, e.g. write SQN=33 as 1_1:

client A got 1_1, 2_1, 3_1, 4_1, 5_1. Authenticates with 1_1.
client B ends up in the same slot, gets 6_1, 7_1, 8_1, 9_1, 10_1.

B authenticates with 6_1. Now {2..5}_1 are invalidated.

A wants to authenticate with 2_1, needs AUTS and gets {11..15}_1.
Uses 11_1, now in turn {7..10_1} are invalidated.

B wants to authenticate with 7_1, needs AUTS and gets {16..20}_1.

And so on. Each client invalidates the remaining tuples of the other client, if they alternate they will need to do AUTS every time, dropping most tuples on the floor.

This will happen for each and every subscriber using clients A and B, so the load increase on the HLR could be substantial.
Admittedly still not noticeable in small setups like ours though...

> Maybe to get the same buckets after reconnection, sort the clients by id... or hash the id to a number 0..31, with a special check against collision... nah whatever...

I think we shouldn't spend too much time to be tricky here for now. Hashing the ID (which might be a stringified GT later on) should be sufficient.

It's not very likely that we will surpass 32 MSCs and SGSNs any time soon, but it would be a bit stupid to have exactly two clients that by coincidence end up getting the same slot due to the hash. Then both would AUTS every time even though there is lots of slot space available. With PS and CS connections usually running at the same time, alternating auth is not unusual.

I think changing over to another bucket once is better than AUTSing all the time. Adding a divertion to an unused slot in case of collision shouldn't be too tricky, but let's start out with the hash then.

> Yes, SEQ should be incremented, not SQN. But I think the specs (33.102 Sect. 6.3.5) state it quite clearly what needs to be dne, i.e., we have to use the SQN provided in the AUTS.
>
> "If the verification is successful the HE/AuC resets the value of the counter SQN HE to SQN MS."

Just to reiterate, SQN.HE in 6.3.5 means the overall largest used SQN per USIM.

We still offset from that to reach the right bucket...

AUTS provides the SQN.MS, the last highest SQN the USIM has used successfully.
We set SQN.HE = SQN.MS, but this may well index to another bucket.
So we increment SEQ and add the index to get to the desired bucket:

SQN.next = ((((SQN.HE = SQN.MS) >> IND) + 1) << IND) + i
(i being the bucket index as in 33.102)
And, finally, set SQN.HE = SQN.next in the HLR db.

VLR subscribers:

> [...] definitely
> worth adding some feature / wishlist tickets at low priority.

#1973#1974

#21 - 03/08/2017 10:08 PM - laforge

- Related to Feature #1969: add IND for proper UMTS auth resync added

#22 - 03/08/2017 10:08 PM - laforge

- Related to Bug #1968: upon auth resync with osmo_auth_gen_vec_auts(), use MS.SQN + (2 ^ IND) added

#23 - 03/08/2017 10:08 PM - laforge

- Related to Feature #1711: 3G Auth added

#24 - 04/25/2017 01:57 PM - laforge

- Status changed from Resolved to Closed